Appendix II

II

DB2:
Concepts, Design, and Programming

JAMES MARTIN

with
Kathleen Kavanagh Chapman
Joe Leben

PRENTICE HALL, Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data

Martin, James
DB2, concepts, design, and programming.

"A James Martin book."
Includes index.
1. Data base management. 2. IBM Database 2
(Computer system) I. Chapman, Kathleen Kavanagh.
II. Leben, Joe. III. Title.
QA76.9.D3M363 1989 005.75'65 88-30727
ISBN 0-13-198581-7

Editorial/production supervision: *Kathryn Gollin Marshak and Karen Skrable Fortgang*
Jacket design: *Bruce Kenselaar*
Manufacturing buyer: *Mary Ann Gloriande*

Printed in the United States of America

10 9 8 7 6 5 4 3

ISBN 0-13-198581-7

PRENTICE-HALL INTERNATIONAL (UK) LIMITED, *London*
PRENTICE-HALL OF AUSTRALIA PTY. LIMITED, *Sydney*
PRENTICE-HALL CANADA INC., *Toronto*
PRENTICE-HALL HISPANOAMERICANA, S.A., *Mexico*
PRENTICE-HALL OF INDIA PRIVATE LIMITED, *New Delhi*
PRENTICE-HALL OF JAPAN, INC., *Tokyo*
SIMON & SCHUSTER ASIA PTE. LTD., *Singapore*
EDITORA PRENTICE-HALL DO BRASIL, LTDA., *Rio de Janeiro*

for it. As a data set reaches its maximum size, DB2 automatically creates a new data set. All the data sets that make up a database are contained in the storage group or groups associated with the table space.

A table space can be altered with an ALTER TABLESPACE statement. The storage group, primary quantity, secondary quantity, erase option, buffer pool, lock size, close value, free space, or password can be changed via the ALTER statement. Other changes must be made by dropping the table space and re-creating it. When a table space is dropped, any tables it contains, and their associated indexes, are lost.

DEFINING TABLES

Tables are created with a CREATE TABLE statement. As we have seen in previous examples, the CREATE TABLE statement gives a name to the table. If the name in the CREATE TABLE statement is not qualified with an authorization ID, DB2 qualifies it with the authorization ID of the table's creator. The CREATE table also assigns names to the columns that make up the table. Referential constraints can be defined by specifying a primary key and/or one or more foreign keys, as discussed in Chapter 6. The statement also specifies data type, length, use of nulls, and use of default values for each column. Here is a CREATE TABLE statement that might be used to create the Suppliers table:

```
CREATE TABLE SUPPLIERS
       (SUPSUPP    SMALLINT NOT NULL,
        NAME       CHAR (15),
        ADDRESS    VARCHAR (35),
        CODE       SMALLINT)
      IN DATABASE TABLE1D3
```

The IN clause can be used to specify a database, as shown in the example. In this case, a table space with the same name as the table is created in the specified database. If another table space in the database already uses that name, the table space is given a name that is a modified version of the table name. The IN clause can also specify both a database name and a table space name:

```
IN INVDB.INVSPACE
```

If the CREATE TABLE statement does not include an IN clause, the table space for the database is placed in the default database, DSNDB04.

Exit Routines

A CREATE TABLE statement can specify the use of exit routines. Exit routines can be used to perform validation and editing on table data. Three types of exit routines can be specified in a CREATE TABLE statement: *validation routines, edit routines,* and *field procedures.*

A validation routine applies to the table as a whole and is specified by including a VALIDPROC clause in the CREATE TABLE statement. The specified validation routine is invoked each time a row is to be updated, inserted, or deleted. A validation routine processes an entire row of data and is used to ensure that the data in the row is valid before the update operation is performed. If the validation routine returns a nonzero return code, the insert, update, or delete operation is not performed.

An edit routine also applies to the table as a whole and is specified by including an EDITPROC clause in the CREATE TABLE statement. An edit routine also processes an entire row of data and is used to encode or decode the row in some way. Possible encode-decode functions include data compression and encryption. The encode function is performed when a row is being inserted or updated; the decode function is performed whenever a row is retrieved.

A field procedure applies to an individual column and is specified by including a FIELDPROC clause in the definition of that column. A field procedure is used to perform encoding and decoding on a single data item. A typical use of a field procedure might be to transform a data item value in a way that will alter its sorting sequence, perhaps because the standard sorting sequence will not produce the desired result. Field encoding is performed whenever that data item is inserted or updated, and field decoding is performed whenever the data item is retrieved.

ALTERING AND DROPPING TABLES

A table's definition can be modified by issuing an ALTER TABLE statement. An ALTER statement allows new columns to be added, primary and foreign keys to be added or dropped, and different exit routines to be specified. Any other changes, such as changing the type or length of an existing column, must be made by first dropping the table, re-creating it, and reloading it. When a table is dropped, any data it contains is lost, and any views or indexes defined on the table are also dropped. If a table is in a partitioned table space, the table cannot be dropped. Instead, the table is deleted by issuing a DROP statement for the table space.

Whenever a table is created, altered, or dropped, the entire database containing the table is locked. For this reason, a user that is granted CREATE, ALTER, or DROP table privileges may be given a private database to use so that other users are not adversely affected by these operations.

DEFINING INDEXES

An index is the mechanism used with DB2 to identify the key of a table. With DB2, a key is a column or set of columns on which an index is defined. If the key consists of two or more columns, it is known as a *composite key*. An index contains pointers associated

with different key values that are used to retrieve rows, to ensure their uniqueness, or to determine where new rows should be physically stored.

An index is created with a CREATE INDEX statement, as shown in the following example:

```
CREATE INDEX XQUOT
       ON QUOTATIONS (PRICE)
```

This statement gives the index a name (XQUOT), specifies the table on which it is defined (*Quotations*), and specifies the names of the column or columns that make up the key (*Price*). In the example, the XQUOT index does not reference a column that contains unique data item values; there may be several rows in the Quotations table that have the same value for PRICE.

A unique index is defined by including the keyword UNIQUE, as shown in the following example:

```
CREATE UNIQUE INDEX XQUOT2
       ON QUOTATIONS (QUOSUPP, QUOPART)
```

The XQUOT2 index causes DB2 to ensure that there will not be two rows that have the same values for supplier number and part number.

When a table is defined with a primary key, a unique index must be defined for the table using the primary key column or columns as the index key. Although this is not required, IBM recommends that for performance reasons an index should be defined for a foreign key as well. A foreign key may or may not have unique values.

A *cluster index* is used to control where rows are physically stored. When a table has a cluster index, rows are stored as closely as possible in the same physical sequence as the order of their index values whenever the table is loaded or reorganized. A cluster index is specified by including a CLUSTER clause in the CREATE INDEX statement. CLUSTER is also used in creating a partitioned index. A partitioned index specifies how data is to be divided between the various partitions of a partitioned table space. The following is an example of a CREATE statement for a partitioned index.

```
CREATE INDEX XINV1
       ON INVENTORY (INVPART)
       CLUSTER
          (PART 1 VALUES (99),
           PART 2 VALUES (199),
           PART 3 VALUES (999))
```

The value shown for each partition identifies the highest value for *Invpart* that should be stored in that partition.

The CREATE INDEX statement may also specify a storage group to use, primary and secondary allocation quantities, the erase and close options, a buffer pool, a password, and free-space requirements. For a partitioned index,

each partition can
is specified, the de
INDEX can also s
into subpages. An
into 2, 4, 8, or 1
comes the unit of
pages increases sto
use subpages and I

When DB2 p
space in the databa
sets needed to cor
index itself is also
then on, whenever
automatically upda

The storage
buffer pool, close
can be modified u
an index, the inde
application plans t
be dropped directl
also drops the ass

DEFINING VIEW

in Chapter 6:

```
CREATE

AS
```

Since a view defi
is based, a CREA
or any of the othe
ment cannot be is
created. Dropping
ping a view does

each partition can be assigned to a different storage group. If no storage group is specified, the default storage group for the database is used. The CREATE INDEX can also specify whether the pages of the index are to be subdivided into subpages. An index always uses a page size of 4K. Pages can be subdivided into 2, 4, 8, or 16 subpages. If subpages are specified, a single subpage becomes the unit of locking, thus increasing concurrency. However, using subpages increases storage and processing overhead, so the decisions of whether to use subpages and how big they should be must balance these factors.

When DB2 processes a CREATE INDEX statement, it creates an index space in the database containing the corresponding table. DB2 defines the data sets needed to contain the index space. If the table already contains data, the index itself is also created using the data item values found in the table. From then on, whenever DB2 updates the data item value in a key column, it also automatically updates the corresponding value in the associated index.

The storage group, primary quantity, secondary quantity, erase option, buffer pool, close option, free-space requirements, and password for an index can be modified using an ALTER statement. To change other characteristics of an index, the index must be dropped and recreated. If an index is dropped, any application plans that use the index must be rebound. A partitioned index cannot be dropped directly; instead, the partitioned table space must be dropped, which also drops the associated table.

DEFINING VIEWS

A view is created with a CREATE VIEW statement. Here is the CREATE VIEW statement we looked at in Chapter 6:

```
CREATE VIEW MYDATA
       (SUPPNAME, D17PART, PARTNAME, LEADTIME,
        ONHAND, ONORDER, PRICE, TOTALPRICE)
    AS SELECT NAME, QUOPART, PNAME, TIME, ONHAND,
              ONORD, PRICE, PRICE * ONORD
       FROM INVENTORY, QUOTATIONS, SUPPLIERS
       WHERE INVPART = QUOPART
             AND SUPSUPP = QUOSUPP
             AND PNAME IN ('NUT', 'BOLT')
```

Since a view defines a virtual table using the data in the real tables on which it is based, a CREATE VIEW does not need to specify a database, storage group, or any of the other options associated with a physical table. The ALTER statement cannot be issued for a view; to change a view, it must be dropped and recreated. Dropping a view affects only programs that use the view itself; dropping a view does not affect any of the tables on which the view is based.